BitTorrent DHT Extensions for IPv6

Juliusz Chroboczek

BEP:	32
Title:	BitTorrent DHT Extensions for IPv6
Version:	11187
Last-Modified:	2009-11-20 17:16:34 +0000 (Fri, 20 Nov 2009)
Author:	Juliusz Chroboczek <jch at pps.jussieu.fr>
Status:	Draft
Type:	Standards Track
Requires:	5
Content-Type:	text/x-rst
Created:	14-Oct-2009
Post-History:	15-Oct-2009: Promote same ID in IPv6 and IPv4 nodes. Drop values6. Added some rationales 11-Nov-2011: Change want to be a list of flags. Add nodes about MTU and source address selection. Define behaviour of PORT message.

Contents

Abstract

This document describes a set of extensions to the BitTorrent DHT [1] to allow operation over IPv6.

Introduction

The address exhaustion of the IPv4 Internet, and the coincident deployment of Network Address Translation (NAT) devices, severely limits the performance of the BitTorrent protocol by making an increasing number of peers unreachable. The simple and efficient solution to this problem is to deploy BitTorrent over the IPv6 protocol, which does not suffer from address exhaustion.

A number of extensions (mostly undocumented) to the BitTorrent protocol have been defined in order to allow it to operate over IPv6. Unfortunately, these extensions only allow locating IPv6 peers by using trackers or through peer exchange -- but not from the DHT. This document describes the procedures and message formats used in the IPv6 DHT.

Operation

There are two distinct DHTs: the IPv4 DHT, and the IPv6 DHT.

The two DHTs are independent, meaning that no IPv6 data is stored in the IPv4 DHT, and, conversely, no IPv4 data is stored in the IPv6 DHT. A node wishing to participate in both DHTs must maintain two distinct routing tables, one for IPv4 and one for IPv6.

While this is not strictly necessary, a node should use the same node id in both tables. This simplifies implementation and debugging, and slightly increases efficiency in the case where the two DHTs are mostly congruent.

Usually, messages only carry data in the address family implied by the network layer protocol. In the case of the find_nodes and get_peers requests, however, the requestor may optionally request that the reply should contain IPv4 node information, IPv6 node information, or both.

Single-protocol nodes

A single-protocol node (IPv4 or IPv6) only participates in one DHT. Such a node does not need to explicitly request data in a given family -- the data provided by default is exactly what it needs.

Dual-stack nodes

While conceptually equivalent to a pair of single-protocol nodes, a dual-stack (IPv4 and IPv6) node can make bootstrap faster and improve the reliability of the protocol by "leaking" data between the two DHTs, as described in the following sections.

Bootstrapping

A dual-stack node that is bootstrapping its DHT will send find_nodes requests in order to populate its routing tables. In order to accelerate this process, it should request both IPv4 and IPv6 node information.

Steady-state operation

Once bootstrap is successful, a node should normally only request IPv4 node information from IPv4 nodes, and IPv6 node information from IPv6 nodes. Requesting node information for both address families in all requests is unlikely to provide useful information, and hence increases network traffic with no benefit.

In order to increase the reliability of the DHT after a single-protocol network outage, however, a node may *ocasionally* send a request for IPv4 node information to an IPv6 node, or a request for IPv6 node information to an IPv4 node.

Maximum packet size

A node obeying this specification must not send UDP datagrams with a payload larger than 1024 octets, and must be able to receive UDP datagrams with a payload of 1024 octets or less. (In the interest of robustness, a node should be able to receive datagrams with larger payloads.)

> **Rationale**: a UDP datagram with a payload of 1024 octets easily fits within the IPv6 minimum maximum packet size, which is 1280 octets [3]. In particular, such a packet will cross a Teredo tunnel with no fragmentation.
>
> In principle, IPv4 nodes might be unable to reassemble packets larger than 576 octets [4]; in practice, however, due to the predominance of Ethernet networks, all IPv4 nodes are able to reassemble packets up to 1500 octets.

Source address selection

The IPv6 DHT should use a socket bound to one of the host's global unicast IPv6 addresses rather than the "unspecified address" (::/128). When selecting to which address to bind, Teredo addresses (addresses in 2001:0000::/32) should be avoided if other global unicast addresses are available.

> **Rationale**: The DHT relies on the publicly visible IP address of each node to remain constant. Since multi-homing is common in IPv6, it is particularly important to bind the IPv6 socket to a well-defined address.
>
> While Teredo traffic can, in principle, be more efficient than native traffic, especially when speaking to other Teredo hosts, experience shows that Teredo routing tends to be brittle; hence, Teredo addresses should be avoided whenever possible.
>
> The same issues apply to multi-homed IPv4 hosts. However, in IPv4 multi-homing is not as common as in IPv6, and in the presence of NAT the issues are somewhat more complicated, so this specification refrains from making any recommendations about binding of IPv4 sockets.

Changes to the BitTorrent protocol

The PORT message, as defined in BEP-5, is extended to work over both IPv4 and IPv6. The information provided by the PORT message only applies to the address it was sent from; this implies that a PORT message sent over IPv4 only advertises participation in the IPv4 DHT, and a PORT message sent over IPv6 only advertises participation in the IPv6 DHT.

Multihomed hosts should take care to only send PORT messages over connections established from the address on which they participate in the DHT.

> **Rationale**: in the presence of the LTEP extension negotiation protocol [2], which advertises a peer's addresses across address families, it would in principle be possible to use the PORT message for both address families. However, since an implementation need not participate in both DHTs, nor use the same port in both DHTs, this specification leaves the role of bridging the two DHTs to the 'find_nodes' message (see below).

Changes and extensions to existing messages

Changes to existing parameters

values

In a reply sent over IPv4, the "values" parameter contains a list of strings, each of which contains compact format IPv4 contact information for a single peer.

In a reply sent over IPv6, "values" contains a list of strings, each of which contains compact format IPv6 contact information for a single peer.

Implementations of this specification must be able to properly parse hybrid "values" lists -- lists containing an arbitrary mixture of 6-octet IPv4 values and 18-octet IPv6 values. However, implementations should not send such hybrid lists, and must not send hybrid lists in a reply to an IPv4 request that doesn't contain a "want" parameter.

> **Rationale**: a request sent over IPv4 with no "want" parameter could originate from a node that implements plain BEP-5, and which might therefore be unable to parse a hybrid list.

New parameters

nodes6

The "nodes6" parameter is analogous to the "nodes" parameter: when present, it carries a string containing the compact IPv6 node information for the 8 closest good nodes in the sending node's IPv6 routing table. This parameter is allowed in replies to the find_nodes and get_peers messages (see below).

want

The "want" parameter is allowed in the find_nodes and get_peers requests, and governs the presence or absence of the "nodes" and "nodes6" parameters in the requested reply. Its value is a list of one or more strings, which may include

- "n4": the node requests the presence of a "nodes" key;
- "n6": the node requests the presence of a "nodes6" key.

For future extensibility, other strings may be present in the list, and must be silently ignored on reception.

> **Rationale**: the "want" parameter is not intended to carry random sundry flags, which can simply be included in the top-level dictionary of the message. Extending the "want" parameter without good reason is not recommended.

Changes to message semantics

find_nodes and get_peers

A node sending a find_nodes or get_peers request may include a "want" parameter containing one or both of the strings "n4" or "n6". A node replying to a find_nodes or get_peers request that includes a "want" parameter should include a "nodes" parameter if the request's "want" parameter contained the string "n4", and should include a "nodes6" parameter if the request's "want" parameter contained the string "n6".

In the absence of a "want" parameter, the reply should include "nodes" if the request was sent over IPv4, and should include "nodes6" if the request was sent over IPv6.

> **Rationale**: this is an incompatible change to the protocol defined in BEP-5, which specifies that "nodes" and "values" are mutually exclusive. However, this change makes the DHT more reliable, and has been deployed by most implementations with no negative effects.

When a node receives a get_peers request and it has peer contact information for the matching address family and info-hash, it should additionally include a "values" parameter containing a list of 6-octet strings if the request was sent over IPv4, and a list of 18-octet strings if the request was sent over IPv6.

A reply sent over IPv4 should not contain 18-octet IPv6 contact information, and a reply sent over IPv6 should not contain 6-octet IPv4 contact information. In other words, the "want" parameter only governs the presence of the "nodes" and "nodes6" parameters, not the interpretation of "values".

> **Rationale**: if the requesting node is a single-stack node, it has no interest in values of the other address family. If the requesting node is a dual-stack node, then it must perform the two announces in parallel; providing both sets of data in both sets of replies merely increases the amount of traffic without giving any extra information.

announce_peers

The syntax of the announce_peers request and reply are unchanged. This implies that an announce_peers request sent over IPv4 may only advertise an IPv4 address, and an announce_peers request sent over IPv6 may only advertise an IPv6 address.

Acknowledgements

I gratefully acknowledge the help of *The 8472* and *arvid* in developing this specification.

References

[1] BEP_0005. DHT Protocol. (http://www.bittorrent.org/beps/bep_0005.html)

[2] BEP_0010. Extension Protocol. (http://www.bittorrent.org/beps/bep_0010.html)

[3] RFC 2460. Internet Protocol, Version 6 (IPv6) Specification. S. Deering, R. Hinden. December 1998.

[4] RFC 791. Internet Protocol. J. Postel. September 1981.

Copyright